移动云环境面向多重服务选择的计算卸载算法 *

何远德 [1]，黄奎峰 [2+]
(1. 西南民族大学 语言实验教学中心，成都 610041; 2. 重庆三峡医药高等专科学校，重庆 404120)

摘　要：移动云计算可以通过计算卸载改善移动设备的能效和应用的执行延时。然而面对云端的多重服务选择时，计算卸载决策是 NP 问题。为了解决这一问题，提出一种遗传算法寻找计算卸载的最优应用分割决策解。遗传种群初始化中，算法联立预定义和随机染色体方法进行初始种群的生成，减少了无效染色体的发生比例。同时，算法为预定义的预留种群设计一种特定的基于汉明距离函数的适应度函数，更好地衡量了染色体间的差异。种群交叉中分别利用近亲交配与杂交繁育丰富了种群个体。算法通过修正的遗传操作减少了无效解的产生，以更合理的时间代价获得了应用分割的最优可行解。应用现实的移动应用任务图进行仿真实验评估了算法效率。评估结论表明，所设计的遗传算法在应用执行能耗、执行时间以及综合权重代价方面均优于对比算法。

关键词：移动云计算；能效；计算卸载；应用分割；执行延时
中图分类号：TP393　　doi: 10.19734/j.issn.1001-3695.2018.12.0876

Offloading decision algorithm for multiple service selection in mobile cloud environment

He Yuande[1], Huang Kuifeng[2+]
(1. *Language Experiment Teaching Center*, *Southwest Minzu University*, *Chengdu* 610041, *China*; 2. *Chongqing Three Gorges Medical College*, *Chongqing* 404120, *China*)

Abstract: Mobile cloud computing can use the computation offloading to improve the energy efficiency of mobile devices and the execution delay of applications. However, in the face of the multiple service selection from cloud, the computation offloading decision is a NP problem. In order to address this problem, a genetic algorithm is designed to find the best application partitioning decision solution for computation offloading. In genetic population initialization, our algorithm combines of predefined and random chromosomes to intialize the population, which can reduce the generation of the inefficient chromosomes. At the same time, the algorithm designs a specific fitness function based on Hamming distance function for the predefined reserved populution, which can better measure the difference between chromosomes. The population crossover uses respectively the inbreeding and crossbreeding to enrich individual species. Our algorithm uses the modified genetic operations to reduce the ineffective solutions and obtain the best feasible solution in a reasonable time. We evaluated the efficiency of the proposed algorithm using graphs of real mobile applications in simulation experiments. The evaluated conclusions denote that our designed algorithm has a better performance than the comparision algorithms on the application execution energy, the execution time and the overall weight cost.

Key words: mobile cloud computing; energy efficiency; computation offloading; application partitioning; execution time

0　引言

移动设备（如智能手机）已经成为人们日常生活中不可缺少的一部分[1]；同时，人们对于在移动设备上所执行的应用的类型的复杂程度也变得更加多样和更加复杂，此时的用户需求不仅包括单一的非计算密集型应用（如简单的算术求解），还包括许多计算密集型应用（如图像处理、视频流处理）。尽管移动设备的可用资源（电池容量、处理能力和内存容量）增加很快，但对于计算密集应用仍然是巨大的障碍[2,3]。借助于云计算技术，移动设备可以按需的提供应用服务请求至云端服务器，该类应用催生了移动云计算的发展[4,5]。移动云计算中，移动设备可以通过将计算密集型任务迁移至功能更强大的云端执行来改善自身的可用资源限制，即应用卸载技术[6~8]。由于受网络条件和云端服务器资源可用性的影响，整个应用卸载至云端并非总是有利的，很多研究正集中于如何选择卸载的部分应用任务至云端执行，以实现移动设备更低的能耗和应用更高的性能增益[9,10]。

多数已有的移动云计算集中于研究单站点的卸载决策[11,12]，即将移动设备上部分应用任务卸载至单一云端服务器。研究表明[13]，在云端多站点环境下进行应用卸载将具有更好的性能。然而多站点环境下的应用卸载是个 NP 问题，求解时间复杂度较高。为了解决这一问题，本文设计一种多重服务选择环境下的基于权重代价的卸载决策模型。模型将应用的执行时间和执行能耗综合地考虑到权重代价目标中，并设计了一种遗传算法对卸载问题进行了求解。本文在遗传算法的设计上作了以下修正：利用预留种群丰富了染色体的多样性，通过修正的选择和交叉操作增加了获得更好卸载解的概率。

收稿日期：2018-12-05；**修回日期：**2019-01-24　**基金项目：**四川省科技厅项目（2016ZR0149）；中央高校基本科研业务费专项资金资助项目（2016NZYQN4）

作者简介：何远德（1977-），男，四川广元人，硕士，实验师，主要研究方向为信息安全与计算机网络；黄奎峰（1974-），男（通信作者），重庆奉节人，讲师，主要研究方向为计算机网络(e_hkf@126.com).

1　相关研究

相关研究中，文献[14]中的 MAUI 是一种移动云中改进能耗的卸载算法，算法利用整型线性规化方法求解了最优卸载决策，但仅仅考虑了能耗优化问题。文献[15]中设计的 ThinkAir 是一种按需的资源分配和卸载模型，同样仅从能耗方面考虑了卸载决策问题。文献[12]提出了一种基于计算卸载的遗传算法，利用遗传优化对服务工作流的卸载问题进行了求解，同样将执行时间和能耗考虑到遗传适应度函数中，实现了最小化的卸载代价。文献[11]将网络带宽因素考虑到卸载决策中，也实现了时间和能耗的双目标优化。文献[16]以满足延时约束最小化能耗为目标设计了卸载决策算法，可以得到应用分割后至云服务器的映射解。文献[17]以约束最短路径为基础设计了随机无线信道环境下的自适应卸载决策，实现了移动设备能耗的优化。文献[18]则均衡考虑能耗、通信延时、全局应用执行时间及应用子任务间的顺序约束等条件，设计了联合调度与卸载算法。以上工作均是在单站点即单服务器环境下的卸载决策求解方法，不能应用于目前云服务多供应方的卸载环境。

文献[13]提出一种静态的多站点卸载算法，但忽略了部分运行参数，如网络带宽变化和能耗变化，其结果与实际环境显得不符。文献[19]提出一种基于分支限界法的多站点卸载决策算法，该考虑虽然考虑了带宽和能耗变化，但在求解中等或大规模应用卸载问题时无法在有效时间内得到最优卸载决策，即算法作出卸载决策的时间远大于在本地移动设备上执行应用的时间。文献[20]利用蚁群优化设计一种 MMRO 算法，可得到多站点时的最优卸载解。该算法综合考虑了决策过程中的收益和风险因素，拥有较好的可行性。尽管以上基于多站点多服务选择的卸载决策算法具备一定可行性，但相关工作或者忽略了影响卸载决策的部分因素，或者在求解卸载决策时的时间复杂度过高，导致其可行性不高。本文将综合考虑移动应用的执行时间和移动设备上的能量消耗，通过更好的遗传操作对多服务选择环境时的应用卸载决策问题进行求解，并与相同类型的算法作出性能比较，证明其优越性。

2　系统模型与问题形式化

本章对应用执行时的卸载决策问题中的全局执行时间、执行能耗以及权重代价模型作出描述。图 1 描述的是一种多服务供应的卸载模型，云端的服务器为具有异构处理能力的资源供应方。表 1 给出本文相关符号说明。

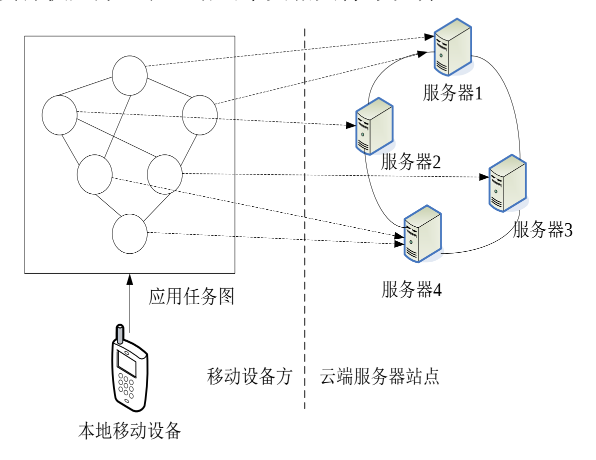


图 1　系统模型

Fig. 1　System model

表 1　符号说明

Table 1　Symbol description

符号	参数含义	符号	参数含义
$xs\ i$	任务 i 的执行因子，若本地执行，$xs\ i$=0，若云端执行，$xs\ i$=1	P^{trans}	本地移动设备的传输功率
k_{xy}	任务间的通信因子	SF^x	云端比较本地设备能力的加速比
X	应用分割解，包括人 p 个子任务，$X=\{x_1,x_2,\ldots,x_p\}$	F_{rp}	预留种群的适应度函数
X'	最优应用分割解	$H(x,y)$	汉明距离函数
$tloc\ i$	应用任务 i 在本地移动设备上的执行时间	LF_{op}	初始种群的局部度函数
T_{local}	应用的本地执行时间	LF_{rp}	预留种群的局部度函数
E_{local}	应用的本地执行能耗	ins_num_i	应用任务 i 的执行指令数
$trem\ i$	应用任务 i 的云端执行时间	$proc_speed^{CPU}$	本地移动设备的 CPU 处理速度
$tx,y\ i,j$	任务 i（分配于云端 x）与 j（分配于云端 y)间的通信时间	$g_x(Cr_i)$	返回初始种群染色体 i 的基因 x 的函数
$ex,y\ i,j$	任务 i（分配于云端 x）与 j（分配于云端 y)间的通信能耗	$g_x(Cr_r)$	返回预留种群染色体 i 的基因 x 的函数
$data_{i\to j}$	应用任务 i 与任务 j 间的通信传输数据量	$g_{i,k}$	表示初始种群染色体 i 的基因 k 的参数
$bwtrans\ x,y$	云端 x 与云端 y 间的传输带宽	$g_{r,k}$	表示预留种群染色体 i 的基因 k 的参数
P^{CPU}	本地移动设备的 CPU 功率		

2.1　应用构成

将执行的应用建立为一个带有权重的任务关系有向图，表示为 $G=(N,E)$，节点 N 和有向边 E 分别表示应用的任务和任务间的联系。假设现有 M 个云服务器集合，集合 $S=\{S_0,S_1,\ldots S_M\}$表示可用服务集合，S_0 表示本地移动设备，即应用任务可在本地移动设备上执行。N 中的每个节点拥有一个权重集合$\{WN0\ i, WN1\ i,\ldots,WNM\ i\}$，$WNx\ i$ 表示在云端 S_x 上执行应用任务 i 的执行时间。同时，每条有向边 $e=(n1,n2)$ 也拥有一个权重集合$\{WE0,0\ \ n1,n2,\ WE0,1\ \ n1,n2,,\ldots,WE0,\mathrm{M}\ n1,n2,WE1,0\ n1,n2,\ldots,WEM,M\ n1,n2\}$，$WE\mathrm{x,y}\ n1,n2$ 表示任务 $n1$（运行于 S_x）与任务 $n2$（运行于 S_y）间的数据传输时间。

2.2　多重服务选择的卸载

2.2.1 权重代价模型

将多重服务选择的卸载问题建立为一个权重优化模型，并寻找应用的最优分割 X'，X'=argmin($W(X)$)。分割 X 的应用权重代价 $W(X)$表示为

$$\min_{w_t,w_e\in[0,1]} W(X) \tag{1}$$

$$W(X)=w_t\times\frac{T(X)}{T_{Local}}+w_e\times\frac{E(X)}{E_{Local}} \tag{2}$$

其中：$T(X)$和 $E(X)$分别表示一个应用的执行时间和执行能耗；T_{Local} 和 E_{Local} 分别表示应用在本地设备上的执行时间和执行能耗；执行时间权重 w_t 和执行能耗权重 w_e 的大小取决于用户应用的执行需求，w_t=1 和 w_e=0 时表示执行时间的优化模

型，w_t=0 和 w_e=1 则表示执行能耗的优化模型。

2.2.2 执行时间模型

应用的执行时间主要由每个任务的执行时间和任务间的通信时间构成。每个任务的执行时间即节点权重分为在本地移动设备上的执行时间 *tloc i* 或云端执行时间 *trem i*，通信时间即为不任务间有向边的权重。执行时间优化模型的目标是寻找应用的最优分割 *X'*，*X'*=argmin(*T*(*X*))。分割 *X* 下应用的执行时间 *T*(*X*)定义为

$$T(X)=\sum_{i=1}^{p}((1-x_i^s)\times t_i^{loc}+x_i^s\times t_i^{rem})+\sum_{i=1}^{p}\sum_{j=1}^{p}k_{xy}\times t_{i,j}^{x,y} \tag{3}$$

其中：*p* 表示应用分割出的任务数量，且

$$x_i^s=\begin{cases}0,s=S_0\\1,s\in\{S_1,S_2,\ldots,S_M\}\end{cases} \tag{4}$$

$$k_{xy}=\begin{cases}0,x_i^s\oplus x_j^s\neq 1\\1,x_i^s\oplus x_j^s=1\end{cases} \tag{5}$$

以上的约束条件表明，若任务分配至云端，则 *xs i*=1；若任务分配在本地设备执行，则 *xs i*=0，*xs i* 可视为本地或远程云端执行的区别因子。k_{xy}代表两个任务间的通信时间因子，若任务被分配至同一云端，则 k_{xy}=0；同时，若两个任务运行于相同移动设备上，k_{xy}=0。此外，式(3)还存在每个任务仅可分配至一个云端服务器的约束，即

$$\sum_{s=0}^{M}x_i^s=1,\forall i\in N \tag{6}$$

每个任务的执行时间需要根据其分配的云端服务器进行计算。因此，每个任务的本地执行时间和远程云端执行时间可分别以式(7)和(8)计算。

$$t_i^{loc}=t_i^x=\frac{ins_num_i}{proc_speed^{CPU}},x=S_0 \tag{7}$$

其中：ins_num_i 表示应用任务 *i* 的执行指令数；$proc_speed^{CPU}$ 表示本地移动设备的处理速度。任务的远程云端执行时间 *trem i* 为其本地执行时间 *tloc i* 除以云端加速因子 SF^x，参数 SF^x 用于描述云端服务器 *x* 比较移动设备处理速度 $proc_speed^{CPU}$ 的加速比。

$$t_i^{rem}=t_i^x=\frac{t_i^{loc}}{SF^x},x\neq S_0 \tag{8}$$

两个任务间的通信时间 *tx,y i,j* 根据任务间的数据传输量 $data_{i\to j}$ 和传输带宽 *bwtrans x,y* 决定：

$$t_{i,j}^{x,y}=\frac{data_{i\to j}}{bw_{x,y}^{trans}} \tag{9}$$

2.2.3 能耗模型

应用的执行能耗主要由每个任务的执行能耗和任务间的通信能耗构成。执行时间优化模型的目标是寻找应用的最优分割 *X'*，*X'*=argmin(*E*(*X*))。分割 *X* 下应用的执行时间 *E*(*X*)定义为

$$E(X)=\sum_{i=1}^{p}((1-x_i^s)\times e_i^{loc}+x_i^s\times t_i^{rem})+\sum_{i=1}^{p}\sum_{j=1}^{p}k_{xy}\times e_{i,j}^{x,y} \tag{10}$$

约束条件为

$$x_i^s=\begin{cases}0,s=S_0\\1,s\in\{S_1,S_2,\ldots,S_M\}\end{cases} \tag{11}$$

$$k_{xy}=\begin{cases}0,x_i^s\oplus x_j^s\neq 1\\1,x_i^s\oplus x_j^s=1\end{cases} \tag{12}$$

$$\sum_{s=0}^{M}x_i^s=1,\forall i\in N \tag{13}$$

任务的本地执行能耗 *eloc i* 和通信能耗 *ex,y i,j* 分别定义为式(14)和 (15)。

$$e_i^{loc}=t_i^{loc}\times p^{CPU} \tag{14}$$

$$e_{i,j}^{x,y}=t_{i,j}^{x,y}\times p^{transfer} \tag{15}$$

其中：p^{CPU} 和 $p^{transfer}$ 分别表示移动设备的 CPU 功率和传输功率。

3 算法设计

本文设计一种基于遗传方法的多重服务选择下的卸载决策算法 ODA-GA。算法中，每个染色体代表应用的一个分割解 *X*，解中的每个任务(基因)被分配一个代表云端服务器或本地设置的值。图 2 代表一个算法中的染色体，此时应用分割为 *p* 个任务，总共有 *M* 个可执行方。ODA-GA 的目标是获得应用的最优分割 *X'*，从而得到最小化的全局执行代价。具体过程如下：

算法 1 ODA-GA

```
a) Initialization//种群初始化
b) Repeat//循环迭代
c) Selection//种群选择
d) Crossover//种群交叉
e) Mutation//种群变异
f) Until the stop criteria are reached//判定迭代结束条件
```

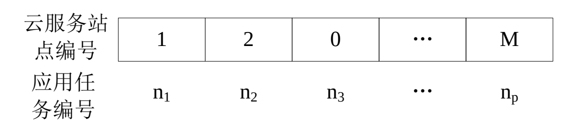


图 2 ODA-GA 算法中的染色体结构

Fig. 2 Chromosome structure of ODA-GA

3.1 初始化

该步骤需要初始化 ODA-GA 算法的相关参数，包括最大迭代次数 *I*、变异概率 *mp*、种群大小 *PS* 及初始种群。多数遗传操作随机选择初始种群，ODA-GA 算法联立预定义和随机染色体方法进行种群初始化。由于移动云环境中进行任务卸载优于本地执行性能，算法创建一个染色体，其所有任务（基因）均分配至本地设备上，这有助于在迭代中避免产生无效染色体。进一步，根据云端服务器的数量 *M*，进行种群初始化，其所有染色体的基因均分配至同一云端服务器，产生 *M* 个染色体。剩余染色体则随机产生，维持种群的随机性。产生初始种群后，ODA-GA 算法利用一个预留种群对染色体进行多样化处理。预留种群包含的染色体不同于初始种群，其染色体的差异（任务的分配目标）较初始种群中更大，有助于获得更高适应度的个体。图 3 为算法的初始种群。

算法 2 Initialization

```
a) set op as original population, rp as reserve population//设置初始种群和预留种群
b) set maximum iteration number I,mutation probability mp//设置最大迭代次数和种群变异概率
c) set population size PS, components' number p and execution sites s//设置种群大小和应用任务数量和执行站点
d) for i =1 to M+1 do//产生预留种群
e) for j=1 to p do
f) op[i][j]=i-1
g) rp[i][j]=random[0,M]
h) end for
i) end for
j) for i=M+2 to PS do//产生随机种群
k) for j=1 to p do
l) op[i][j]=random[0,M]
```

```
m) rp[i][j]=random[0,M]
n) end for
o) end for
```

图 3 ODA-GA 算法的初始种群

Fig. 3 Initial population of ODA-GA

3.2 适应度函数

ODA-GA 算法中，初始种群的适应度函数定义为式(1)，因此，每次迭代中，算法将试图寻找比先前迭代中具有更小权重代价的应用分割解。由于 ODA-GA 算法利用一个预留种群使染色体更多样化，需要根据预留种群的目标定义另一个适应度函数。预留种群的适应度函数 $F_{rp}(r)$为

$$F_{rp}(r)=\frac{1}{PS}\sum_{i=1}^{PS}H(Cr_i,Cr_r) \tag{16}$$

其中：PS 表示初始种群大小；Cr_i 表示初始种群中第 i 个染色体；Cr_r 表示预留种群中的一个给定染色体；H 表示汉明距离函数，定义为

$$H(Cr_i,Cr_r)=\sum_{k=1}^{p}\mathrm{sgn}(|g_{i,k}-g_{r,k}|) \tag{17}$$

其中：

$$\mathrm{sgn}(|g_{i,k}-g_{r,k}|)=\begin{cases}0, x=y\\1, x\neq y\end{cases} \tag{18}$$

$F_{rp}(r)$的汉明距离函数用于计算给定染色体间的差异，根据任务分配目标不同衡量。$g_{i,k}$ 和 $g_{r,k}$ 分别表示初始种群中染色体 i 的第 k 个基因和预留种群中染色体 r 的第 k 个基因。

3.3 种群选择

该步骤中，初始种群和预留种群中的染色体根据其适应度函数进行评估，每个种群的染色体被排序，并根据锦标赛选择法进行染色体选择。每个种群中最优的两个染色体被选择产生子代，选择过程如算法 3 所示。

算法 3 Selection

```
a) For i=1 to PS do
b) op_fitness[i]=calculate fitness on the basis of Equ.1//利用式(1)计算初始种群的适应度
c) rp_fitness[i]= calculate fitness on the basis of Equ.16//利用式(16)计算预留种群的适应度
d) End for
e) Sort op_fitness//基于适应度对初始种群排序
f) Sort rp_fitness//基于适应度对预留种群排序
g) Selected_op=select best two chromosomes of op_fitness//选择初始种群中最优的两个染色体
h) Selected_rp=select best two chromosomes of rp_fitness//选择预留种群中最优的两个染色体
```

3.4 种群交叉

交叉步骤中，所选择的染色体（父代）相互结合产生新的子代。为了利用更加智能的交叉操作，算法分别为初始种群和预留种群定义局部适应度函数 LF_{op} 和 LF_{rp}。

$$LP_{op}(g_i)=w_t\times(t_{gi}^{x}+\sum_{j=1}^{p}k_{xy}\times t_{i,j}^{x,y})+w_e\times(e_{gi}^{x}+\sum_{j=1}^{p}k_{xy}\times e_{i,j}^{x,y}) \tag{19}$$

其中：$tx\ gi$ 和 $ex\ gi$ 分别表示初始种群染色体中第 i 个基因（任务）的执行时间和执行能耗；$tx,y\ i,j$ 和 $ex,y\ i,j$ 分别表示两个基因间的通信时间和通信能耗。式(11)~(13)也为式(19)的约束条件。

$$LF_{rp}(g(Cr_r))=\sum_{i=1}^{PS}H(g_x(Cr_i),g_x(Cr_r)) \tag{20}$$

其中：$g_x(Cr_i)$和 $g_x(Cr_r)$分别表示返回初始种群染色体 i 的基因 x 和预留种群染色体 r 的基因 x 的函数。

由于拥有两个种群，算法利用杂交繁育方法对种群进行交叉操作，即不同种群的父代可以进行交叉，这可以交换不同种群间的信息。同时，同一种群父代染色体可以进行近亲交叉。图 4 显示了 ODA-GA 算法的杂交和近亲交叉示例。

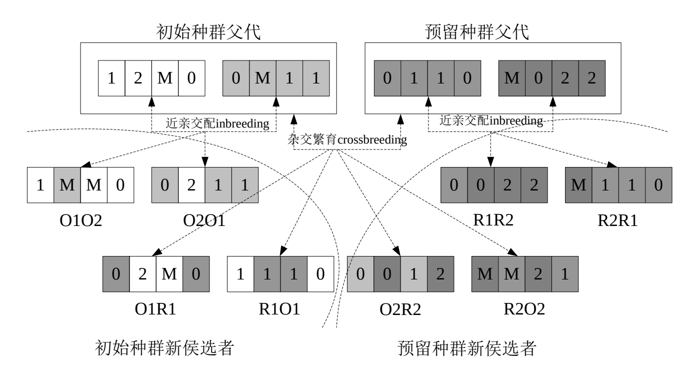


图 4 ODA-GA 算法的交叉实例

Fig. 4 Crossover example of ODA-GA

近亲交配(inbreeding)中，同一种群的父代染色体相互结合，两个父代的基因作出比较，拥有更优局部适应度(LF_{op} 或 LF_{rp})的基因被复制至子代中。同时，剩余基因产生另一子代。杂交繁育(crossbreeding)中，初始种群中的每个父代与预留种群中的父代结合，并基于 LF_{op} 产生两个子代。该步骤基于 LF_{rp} 进行重复产生另外两个子代。因此，在每次迭代中，将产生八个子代。ODA-GA 算法的交叉过程如算法 4 所示。

算法 4 Crossover

```
a) %inbreeding and crossbreeding are crossover functions//种群交叉函数包括近亲交配和杂交繁育
b) %both of them use local fitness to generate new offsping
c) %each one of inbreeding and crossbreeding generate two offspring
d) %Par_op and Par_rp represent chromosomes of original population and reserve population respectively
e) For i=1 to 2 do
f) For j=1 to p do
g) LF_op[i][j]=calculate local fitness by Equ.19//利用式(19)计算初始种群的局部适应度
h) LF_rp[i][j]=calculate local fitness by Equ.20//利用式(20)计算预留种群的局部适应度
i) end for
j) end for
k) original population=inbreeding(Par_op,Par_op)//初始种群的近亲交配
```

l) reserve population=inbreeding(Par_{op},Par_{op})//预留种群的近亲交配

m) original population=crossbreeding(Par_{op},Par_{op})//初始种群的杂交繁育

n) reserve population= crossbreeding(Par_{op},Par_{op})//预留种群的杂交繁育

3.5 种群变异

变异步骤中，新染色体的基因根据在初始化中预先定义的概率进行变异。ODA-GA 算法中，基因根据标准均匀突变模型进行变异，该模型中，所有基因的变异概率是相等的。算法 5 描述 ODA-GA 的变异过程。

每次迭代结束时，由原染色体组成的初始种群和预留种群和新的子代均被排序，且最优染色体在整个种群中得以选择。当到达预定义的迭代次数或无法进行适应度改进时，ODA-GA 算法得以完成。

算法 5 Mutation

```
a) %numoff and mp represent the number of offspring and the mutation probability respectively
b) for i=1 to numoff do
c) for j=1 to p do
d) offspring[i][j]=Random[0,M,mp]
e) end for
f) end for
```

4 实验评估

4.1 实验环境配置

本节通过现实的应用任务图评估 ODA-GA 算法的性能。利用 SPECjvm 实验床[21]中两种开源移动应用 ESS 和 DBAP 进行实验仿真，ESS 应用为一种专家系统 shell 程序应用，DBAP 应用为一种数据库访问程序应用，ESS 应用中总共拥有 20 个任务节点，有向边为 76 条，DBAP 应用中总共拥有 33 个任务节点，有向边为 82 条。算法在 Matlab R2014 中模拟实现，硬件配置为 2.2 GHz Intel core i7CPU，8 GB RAM。配置一个具有三个异构执行站点的多站环境，包括两个云端服务器和一个移动设备作为本地站点。每个云服务器的加速因子服从区间[3,4]内的均匀分布。任务图中的权重同样服从均匀分布，所有算法执行 10 次，获取 95%的置信区间。权重 w_t 和 w_e 均设置为 0.5，即应用执行对于执行时间和执行能耗具有同等的偏好，该权重值可以通过不同的应用类型进行不同配置。

4.2 参数影响

ODA-GA 算法中，种群大小 *PS*、最大迭代次数 *I*、变异概率 *mp* 以及网络带宽 *bwtrans x,y* 均会对算法性能产生影响。为了调整 ODA-GA 算法使其得到最优的性能表现，实验如表 2 所示配置了不同的场景。

表 2 参数配置

Table 2 Parameter setting

配置场景	*PS*	*I*	*mp*	*bwtrans x,y*
C1	10-100	30	0.07	100
C2	50	10-200	0.07	100
C3	50	30	0.02-0.9	100
C4	50	30	0.07	10-100

图 5 表示在两种应用下根据配置场景 C1 得到的种群大小对 ODA-GA 算法性能的影响。随着种群大小从 10 增加至 50，两种应用的权重代价急剧下降。然而进一步增加种群规模并不会对种群适应度的改进产生巨大影响，即在 *PS*=50 后，种群的适应度并未发生明显变化，因此，后文将 50 设置为种群大小。图 6 显示在两种应用下根据配置场景 C2 得到的迭代交数对 ODA-GA 算法性能的影响。可以看到，到达稳定结果时（即最优适应度），迭代次数为 30。继续迭代也未对种群适应度改进产生明显影响。因此，后文将 30 作为算法的迭代次数。图 7 显示在两种应用下根据配置场景 C3 得到的变异概率对 ODA-GA 算法性能的影响。可以看到，变异约为 0.07 时算法产生了最优性能。算法的最差表现时变异概率为 0.6。图 8 显示在两种应用下根据配置场景 C4 得到的带宽对 ODA-GA 算法性能的影响。如预期的，更高的带宽将导致更佳的适应度，由于带宽的增加，可以降低任务的通信延时和通信能耗。网络约增加至 100 KB/s 时权重代价才趋于稳定。

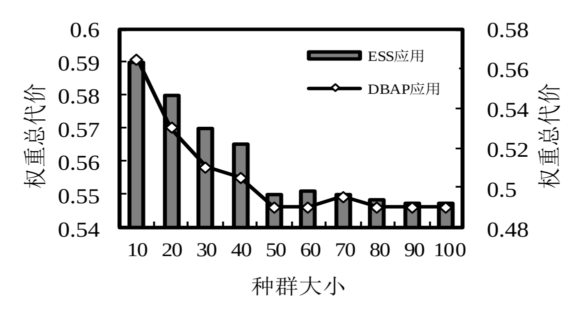


图 5 种群大小对权重代价的影响

Fig. 5 Population size impacts on weight cost

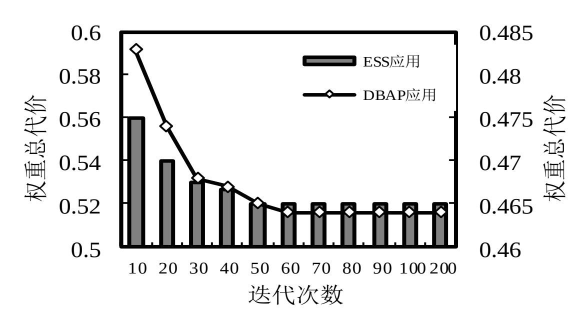


图 6 迭代次数对权重代价的影响

Fig. 6 Iteration number impacts on weight cost

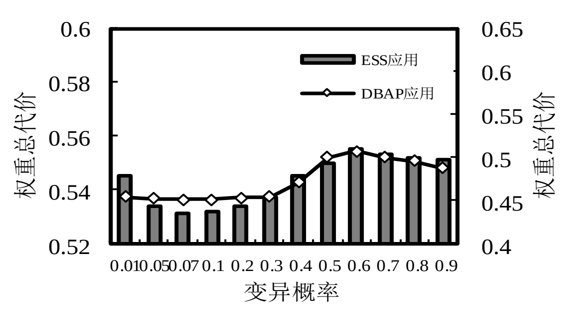


图 7 变异概率对权重代价的影响

Fig. 7 Mutation probability impacts on weight cost

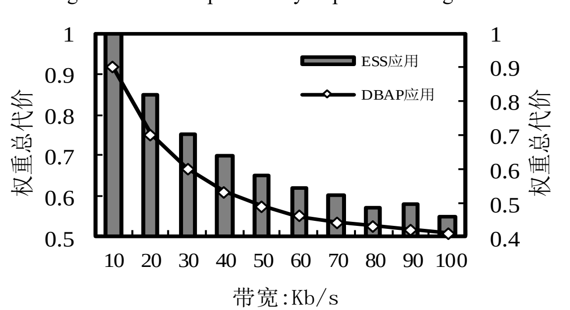


图 8 带宽对权重代价的影响

Fig. 8 Bandwidth impacts on weight cost

4.3 对比研究

本节作算法的对比分析，对比算法包括：

a) Local 算法。该算法中所有应用任务均在本地移动设备上执行。

b) Optimal 算法[19]。该算法利用 branch-and-bound(分支限界法)得到应用分配最优解，当求解模型较大时，该算法求解复杂性过高，因此所得到的解仅用于比较其他算法解的效率。

c) SGA 算法[22]。该算法即为常规的 GA 算法，遗传操作上未作任何修正。

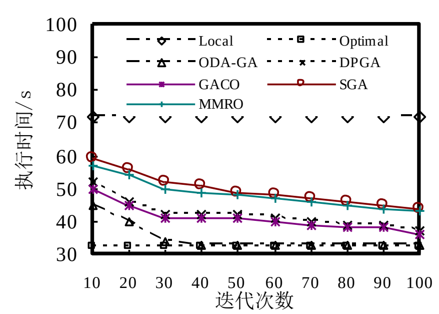
d) DPGA 算法[23]。该算法也利用初始种群和预留种群进行解进化，但其他步骤类似于 SGA。

e) GACO 算法[12]。为单一服务环境下的遗传卸载算法。

f) MMRO 算法[20]。该算法为基于蚁群优化的多站点卸载决策算法。

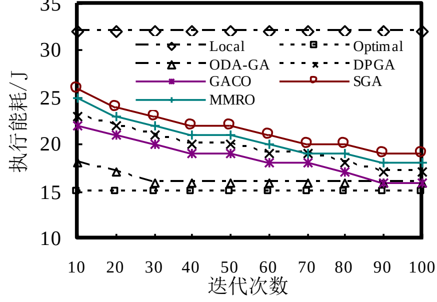
g) Local 算法作为衡量计算卸载是否能够改善性能的标准算法。实验中参数配置由前一部分得到的最优值进行设置，PS=50，I=30，mp=0.07，带宽为 100。

图 9 为迭代次数改变时的实验结果。可以看到，所有基于任务卸载的算法的执行时间和能耗均随着最大迭代次数的增加而下降，ODA-GA 算法比其他算法在三项指标上更加有效，算法可以更少的迭代次数 30 次并以更快的速度收敛到最优解上，这表明该算法的决策时间也更短。MMRO 算法的性能与 SGA 极其相似，而 DPGA 算法也与 GACO 拥有相似的性能。图 10 是带宽改变时的实验结果。ODA-GA 算法在最差情况下（带宽为 10）的性能与局部执行算法时的性能相似。然而 SGA、MMRO、DPGA 和 GACO 等其他算法的性能在最差情况下与局部执行算法相差较远，即使在较低的带宽下，ODA-GA 算法相较对比算法也具有很好的优越性。表 3 给出了算法在 DBAP 应用中算法的决策时间情况。ODA-GA 算法较其他算法获得了最优的适应度，其决策时间与其他算法相差不大。SGA 的决策时间是所有算法中最小的，但其适应度是最差的。在 GACO、MMRO 和 DPGA 中，GACO 在决策时间和适应度方面均是更优的。



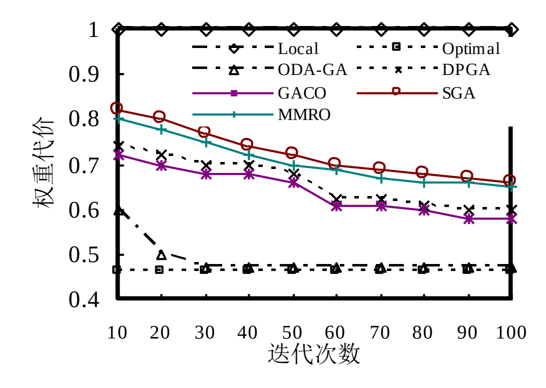
(a)迭代次数对执行时间的影响

(a)Iteration number impacts on execution time



(b) 迭代次数对执行能耗的影响

(b)Iteration number impacts on execution energy

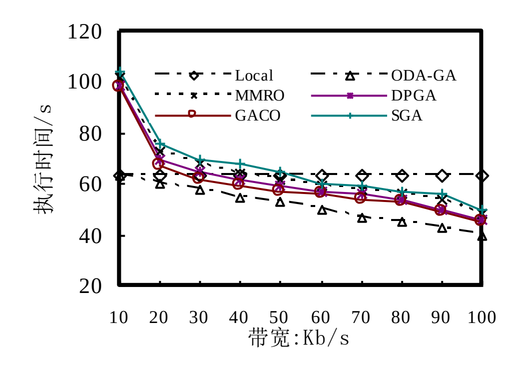


(c) 迭代次数对权重代价的影响

(c)Iteration number impacts on weight cost

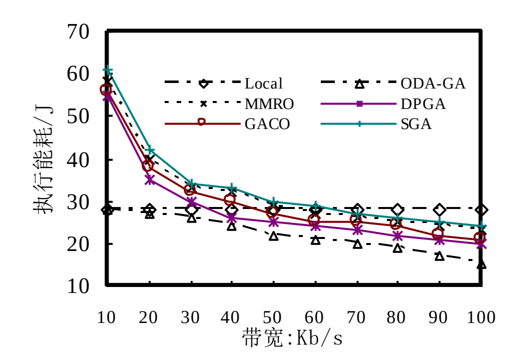
图 9　运行 DBAP 应用时迭代次数对算法的影响

Fig. 9　Iteration number impacts on algorithms in DBAP



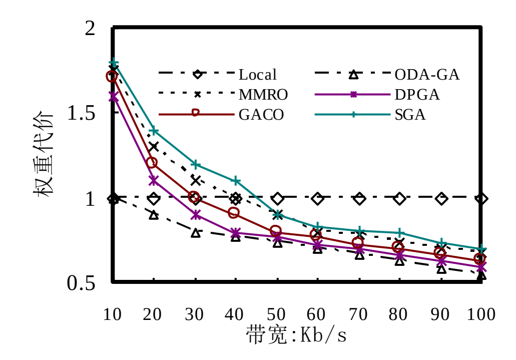
(a)带宽大小对执行时间的影响

(a)Bandwidth impacts on execution time



(b)带宽大小对执行能耗的影响

(b) Bandwidth impacts on execution energy



(c)带宽大小对权重代价的影响

(c)Bandwidth impacts on weight cost

图 10　运行 DBAP 应用时带宽对算法的影响

Fig. 10　Bandwidth impacts on algorithms in DBAP

表 3 决策时间对比

Table 3 Decision time comparison

算法	决策时间/s
ODA-GA	0.47
DPGA	1.05
GACO	0.34
MMRO	0.4
SGA	0.3

5 结束语

为了解决多服务选择环境下的计算卸载决策问题，本文提出了一种遗传算法寻找计算卸载的最优应用分割决策解，算法通过修正的遗传操作减少了无效解的产生，以更合理的时间代价获得了最优可行解。应用现实的移动应用任务图进行了仿真实验评估了算法效率。结果表明，遗传算法在应用执行能耗、执行时间以及权重代价方面均优于对比算法。进一步的研究将在考虑移动设备的移动性和连通性及不同类型的云服务提供的环境下设计高效的卸载决策，在设备执行能耗与应用执行延时上取得更好的均衡。

参考文献：

[1] Li Bo, Liu Zhi, Pei Yijian, *et al.* Mobility prediction based opportunistic computational offloading for mobile device cloud [C]// Proc of IEEE International Conference on Computational Science & Engineering. 2014: 786-792.

[2] Chen Min, Hao Yixue, Li Yong, *et al.* On the computation offloading at Ad hoc cloudlet: architecture and service modes [J]. IEEE Communications Magazine, 2015, 53 (6): 18-24.

[3] Li Yujin, Wang Wenye. Can mobile cloudlets support mobile applications? [C]// Proc of IEEE INFOCOM. 2014: 1060-1068.

[4] Park J S, Kim H, Jeong Y, *et al.* Two-phase grouping-based resource management for big data processing in mobile cloud computing [J]. International Journal of Communication Systems, 2014, 27 (6): 839-851.

[5] Chen Lienwu, Ho Yufan, Kuo Weiying, *et al.* Intelligent file transfer for smart handheld devices based on mobile cloud computing [J]. International Journal of Communication Systems, 2015, 30 (1): 43-53.

[6] Zhou Bowen, Dastjerdi A V, Calheiros R, *et al.* mCloud: a context-aware offloading framework for heterogeneous mobile cloud [J]. IEEE Trans on Services Computing, 2017, 23 (9): 1-12.

[7] Enzai N I M, Tang Maolin. A taxonomy of computation offloading in mobile cloud computing [C]// Proc of IEEE International Conference on Mobile Cloud Computing. [S.l.]: IEEE Computer Society, 2014: 19-28.

[8] Chen Xu. Decentralized computation offloading game for mobile cloud computing [J]. IEEE Trans on Parallel & Distributed Systems, 2014, 26 (4): 974-983.

[9] Liu Jun, Ahmed E, Shiraz M, Application partitioning algorithms in mobile cloud computing: taxonomy, review and future directions [J]. Journal of Network and Computer Applications, 2015, 48 (2): 99-117.

[10] Jeong Y, Park J S, Park J H. An efficient authentication system of smart device using multi factors in mobile cloud service architecture [J]. International Journal of Communication Systems, 2015, 28 (4): 659-674.

[11] Niu Jianwei, Song Wenfang, Atiquzzaman M, *et al.* Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications [J]. Journal of Network & Computer Applications, 2014, 37 (37): 334-347.

[12] Deng Shuiguang, Huang Longtao, Taheri J, *et al.* Computation offloading for service workflow in mobile cloud computing [J]. IEEE Trans on Parallel & Distributed Systems, 2015, 26 (12): 3317-3329.

[13] Sinha K, Kulkarni M. Techniques for fine-grained, multi-site computation offloading [C]// Proc of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. 2013: 184-194.

[14] Cuervo E, Balasubramanian A, Cho D K, *et al.* MAUI: making smartphones last longer with code offload [C]// Proc of IEEE International Conference on Mobile Systems. 2013: 49-62.

[15] Kosta S, Aucinas A, Hui Pan, *et al.* ThinkAir: dynamic resource allocation and parallel execution in the cloud for mobile code offloading [C]// Proc of IEEE INFOCOM. 2013: 21-29.

[16] Barbarossa S, Sardellitti S, Lorenzo P D. Computation offloading for mobile cloud computing based on wide cross-layer optimization [C]// Proc of Future Network & Mobile Summit. 2013: 56-63.

[17] Zhang Weiwan, Wen Yonggang, Wu Dapeng Oliver. Collaborative task execution in mobile cloud computing under a stochastic wireless channel [J]. IEEE Trans on Wireless Communications, 2014, 14 (1): 81-93.

[18] Mahmoodi S E, Uma R N, Subbalakshmi K P. Optimal joint scheduling and cloud offloading for mobile applications [J]. IEEE Trans on Cloud Computing, 2016, 4 (9): 1-10.

[19] Liu Kaiyang, Peng Jun, Li Heng, *et al.* Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing [J]. Future Generation Computer Systems, 2016, 64 (3): 1-14.

[20] Wu Huijun, Huang Dijiang. Modeling multi-factor multi-site risk-based offloading for mobile cloud computing [C]// Proc of IEEE International Conference on Network & Service Management. 2014: 230-235.

[21] Bhattcharya A, De P. Computation offloading from mobile devices: can edge devices perform better than the cloud? [C]// Proc of IEEE International Workshop on Adaptive Resource Management & Scheduling for Cloud Computing. 2016: 1-11.

[22] Dasgupta K, Mandal B, Dutta P, *et al.* A genetic algorithm (GA) based load balancing strategy for cloud computing [J]. Procedia Technology, 2013, 10 (2): 340-347.

[23] Cai Zhiming, Chen Chongcheng. Demand-driven task scheduling using 2D chromosome genetic algorithm in mobile cloud [C]// Proc of IEEE International Conference on Progress in Informatics & Computing. 2014: 539-545.